AUTOMATIC RECOGNITION AND REPRESENTATION OF DOCUMENTS

LYNN D. WILCOX and A. LAWRENCE SPITZ

Xerox Palo Alto Research Center

ABSTRACT

Automatic recognition of the content of documents yields a form of compression for document transmission and archival, and provides a high-level representation with which to manipulate the document. The ease with which a document can be manipulated depends on the representation of the components of the document and on the ability of the recognition system to produce a set of meaningful components. Ideally, a document is segmented into component regions containing text, line drawing graphics and pictures. The text is converted to character codes and related text format information. The line drawing graphics are translated into graphics primitives, and pictures are optionally processed by half-tone screen detection and removal. These outputs are then translated into a document representation language, which also describes the layout structure of the document. In order to study issues in the design and use of document recognition systems, we are building a testbed system. It consists of a text/graphics segmenter, a structure analyzer which groups these segments into higher level concepts, the Kurzweil Discover (a multi-font character recognizer and format analyzer), and a module which translates these outputs into the document representation language, Maker Interchange Format (MIF), used by the Frame Maker publishing system.

1. Introduction

The task of a document recognition system is to convert the scanned image of a document into a representation language which models the original document as accurately and concisely as possible. Automatic recognition and representation of the contents of a document is an alternative to storing the document as a bitmap image. Not only does it greatly reduce the storage requirements, but it allows manipulation of the document in ways which would be impossible with only the image. Alterations of a bitmap document are limited to cut, paste and scale operations. To correct the spelling of a word would be difficult, particularly if the incorrect spelling had too few letters. On the other hand, a document which is described in a high-level language can be modified by editing text and graphics and by moving components to alter the layout. In addition, it allows automatic indexing for filing, storage and retrieval.

Document recognition comprises segmentation and classification of regions of text, line drawing graphics and pictures, and translation of this information into a document representation language. A segmenter classifies possibly overlapping rectangular regions of text and graphics, and a structure analyzer translates these regions into higher level descriptions such as columns of text and lines of text on graphics. Text is translated into its ASCII equivalent, and format characteristics such as font type and size, paragraph breaks, line spacing and centering are detected. Graphics recognition is performed to identify the graphics primitives [Henderson86]. Pictures are processed to detect possible half-tone screening, allowing data compression and optimal rendition on different output devices. These outputs are then translated into a document representation language. In this form, the document can be easily edited, stored or transmitted.

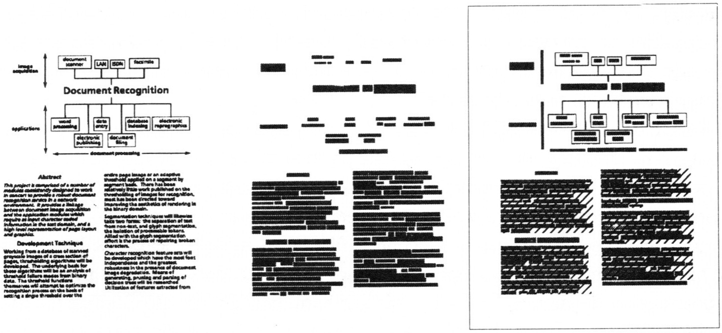
We currently limit ourselves to recognition of the layout structure of the document. That is, we attempt to recognize only the way the components of the document are placed on the page. We do not try to infer the meaning or the logical structure of the document from the set of recognized components. For instance, we do not try to interpret a line of text at the top of a document as a header or title, but simply declare it a line of text at a given location.

Commercial systems exist which provide subsets of the functionality required in a document recognition system. The Kurzweil Discover [Kurzweil87] can convert text to ASCII and recognize certain format characteristics, but cannot automatically handle mixed text and graphics or multi-column text. The GTX [GTX87] vectorization system can perform limited graphics recognition. The Frame Maker [FrameMaker87] publishing system can import text and bitmap graphics, and contains a set of tools for WYSIWYG editing. It also has a language for document representation, the Maker Interchange Format (MIF). However, none of these systems are capable of working together.

Several researchers are investigating aspects of the document recognition problem. A goal of the ESPRIT project HERODE [Meynieux86] is to test the adequacy of the proposed international standard document representation language ODA [Horak85] by using it as the target language for their document recognition system. Kida et al. [Kida86] describe a network-service document recognition system which allows users to edit a recognized file at their workstations, but no mention is made of structure recognition or document representation. A system described in Wong et al. [Wong82], discusses an interactive method for segmentation and character recognition. Srihari and Zack [Srihari86] discuss a method for the segmentation and labeling of document image components.

This paper discusses the current state in our development of a document recognition system. Our design methodology for this system is organized around a cycle of prototyping, experiment and theory development. Our prototype is a modular testbed which allows us to perform experiments while simultaneously working on the

Figure 1: *the original document (a), lines of text (b), and blocks of text, lines of text, and graphics (c)*



theoretical aspects of document recognition. We will describe the following components of our testbed: the segmentation algorithm which separates text and graphics, the structure analyzer which uses this information to infer a higher level description of the layout structure of the document, the integration of the Kurzweil character recognition system and format analyzer, the selection of a document representation language, and the procedure for translating the recognized components into the document representation language.

2. Segmentation and structural analysis

2.1. *Segmentation and classification*

The segmenter identifies rectangular regions of the document and classifies them as either a text line, a text block, or a graphics block. Our current system does not distinguish between line drawing graphics and pictures: all nontext regions are labeled graphics and treated as a bitmap image. A text block is defined as a set of equally spaced lines of text with left justification and approximately equal length. In order for the requirement of equal spacing to be meaningful, a text block must have at least three lines. A text line is an isolated line of text, or a line of text not meeting the text block criteria. The distinction is made between text lines and text blocks because a block of text can be treated as a single flow of text in which line breaks may be ignored, while line breaks in the text lines of a formula or figure label may be critical. The rectangular regions are allowed to overlap, so that text within graphics is supported.

The document is first described in terms of a list of marks (connected black regions). Individual marks are classified as text or graphics based on their size, complexity and classification of neighboring marks. The formation of text lines is based on the linear (horizontal) adjacency of individual marks which have been classified as text. Thus, with knowledge of the distribution of character widths, an inference is drawn about maximal inter-character spacing. In this manner, adjacent characters are clumped into words. Approximately collinear words with appropriate word-spacing are grouped into lines of text, provided there is no intervening graphic structure, such as a rule or grid. Figure 1 shows the lines of text of the document in (a), displayed as shaded rectangles in (b).

Text blocks are formed from lines of text by detecting periodicity in the line spacing. The autocorrelation of the absolute position of the base of each mark is used to estimate the vertical periodicity. Characters which rest on the baseline contain information on periodicity, while characters with descenders add noise to the information. The autocorrelation function is filtered to enhance peaks resulting from the baseline characters while suppressing noise from descenders. The first peak in the autocorrelation function as well as its expected harmonics are used to estimate the line spacing. Allowance is made for noise to corrupt a single isolated harmonic. Lines whose spacing agrees with the average spacing computed in this way are grouped to form a text block.

Graphics blocks are defined as minimal enclosing rectangles surrounding individual marks previously classified as graphics. If graphic blocks are adjacent or intersecting, they are merged to form a new graphics block and the enclosing rectangle is redefined. Figure 1(c) shows the text blocks of the document in Figure 1(a) as diagonally hatched rectangles, the remaining text lines as shaded rectangles and the graphics in its original form.

2.2. *Structural analysis*

The purpose of the structure analyzer is to infer a higher level structure of the document from the recognized components and their positions. While the position of individual text lines, text blocks and graphics blocks is sufficient to describe the layout structure of the document, a higher level description is preferable for integration with the rest of the system. For instance, the interface to the character recognition module is optimized by sending the maximum amount of text possible. In the case of intersecting text and graphics blocks, the concept of being above is important for layout. In our system, a bitmap drawn over text obscures the text, while the reverse is not true. Levy calls this the 2½ dimensionality of documents [Levy87].

The higher-level components of a document are thus columns of text and text on graphics. Columns of text provide the character recognition module with the maximum amount of text, subject to the constraint of single column data with no graphics. The text on graphics component implies that the text is superimposed on the graphics. The concept of text on graphics is also useful in the document representation language, since it allows later manipulation of the text and graphics as a unit.

Figure 2: *a segmentation tree: leaf nodes are text lines and graphics, intermediate nodes are text blocks, columns of text and text on graphics*

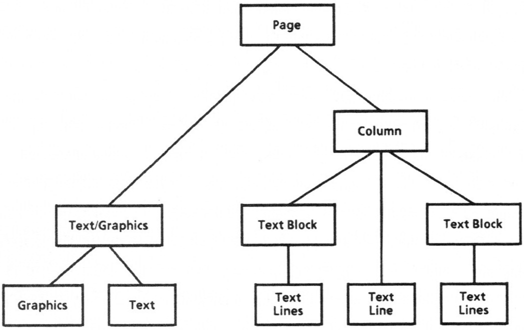


Figure 2 shows the hierarchical representation of a document page. The tree is built from the bottom up, with the segmenter supplying the text lines, text blocks and graphics. Columns of text are built by merging text blocks of approximately equal width (allowing for text set ragged right) that are vertically adjacent and that have no intervening graphics. Lines of text between text blocks, such as titles, should also be included in the column. Therefore, a line of text is redefined as a text block of arbitrary width, and the same rules for vertical adjacency and the exclusion of graphics are applied. Text on graphics is the union of all text blocks or text lines that intersect a given graphics block. The enclosing rectangle for the text on graphics component is redefined as the smallest rectangle enclosing all the text and graphics blocks.

3. Character recognition

The primary job of character recognition is to transform text portions of the scanned document image into their ASCII equivalents. For a system of this type, recognition should be font independent and should require little or no training by the user. By font independent, we mean that the system can recognize most fonts, except stylistic ones, produced by any output device ranging from a typewriter to a laser printer to a printing press. This is in contrast to certain systems, e.g., DEST [DEST87], which recognize only fixed pitch typewriter fonts. User-trained systems require the user to instruct the system as to the identity of each character it sees. Although this mode is useful for applications with large quantities of homogeneous text, we do not see this as practical in a system which expects many different types of documents.

For accurate reproduction of the document, it is also important to recognize the text format characteristics. This includes identification of paragraphs, recognition of font type and size, subscript and superscript detection, recognition of line centering and right justification, and estimation of line spacing and paragraph spacing. Font identification is a difficult task in that it requires examination of detailed characteristics of the text. In fact it is just this level of detail that an omni-font character recognizer wishes to ignore. However, an approximation can be made by recognizing whether the font has fixed or variable spacing.

Work on a font-independent character recognizer based on descriptive features [Spitz87] is in progress. In order to test other components of our system, we are currently using the Kurzweil Discover [Kurzweil87], a multi-font character recognition system which does not require any training by the user. The system performs character recognition and tags questionable and unrecognized characters for ease of later editing. A format analysis mode is available in which format descriptions in the form of escape codes are interspersed with the recognized characters. In addition, a file is generated which gives paragraph profiles for each distinct paragraph type. The paragraph profile file also includes the font descriptions. Our system uses the escape codes to recognize the start and the end of paragraphs, font changes, subscripts, and superscripts. The escape codes also indicate the actual line breaks of the original text and flag hyphens positioned at the ends of lines. We chose to ignore these line breaks in favor of those generated when reproducing the document from the representation language. However, the line break information is used in the translation procedure. The paragraph profiles define the font characteristics of all the fonts used in the paragraph, i.e., point size and whether the font is fixed or variable pitch. The profile also gives the line spacing, spacing above and below the paragraph, and whether the text is right justified or ragged.

4. Document representation language

After the document has been segmented and components of the document have been recognized, the outputs must be translated into the document representation language. This representation should describe the original document as closely as possible in a compressed form. This representation should be such that the document can be easily manipulated by arbitrary editing of text, graphics and layout. If the document is to be shared with other systems, a standard representation for storage and transmission is important.

An international standard for document representation such as ODA [Horak85] is highly attractive. ODA defines simultaneously the logical structure of the document, e.g., titles and paragraphs, and the layout structure of the document, or how it looks on paper [Joloboff86]. Unfortunately, a full implementation of the standard does not yet exist. Another possibility for document representation is SGML [International86], an international standard document description technique developed by IBM. SGML